Evolving Postmortems as Teams Evolve Through TxP

Brad Hodgins, NAVAIR

Abstract. TxP was created by NAVAIR to help non-software teams define their own customized team processes, similar to the TSP process enjoyed by software teams. It takes time for a team to define a customized team process, and during this time, postmortem analysis needs to evolve as the team's customized process evolves. These postmortems will give the team the insight required to see where it has improved, and where it needs to focus its future improvement efforts on its way to standing up its customized team process.

The success that NAVAIR [1] software teams have had using Team Software Process (TSP) had led some of their parent organizations to desire to achieve the same level of performance from their non-software teams. In response to these requests, NAVAIR got together with the Software Engineering Institute (SEI) to develop an approach based upon the same fundamental principles behind TSP (i.e., plan your work, work your plan, and analyze your data), but with no specific domain discussed in the methodology. NAVAIR came away from that development effort with Team Process Integration (TPI), which requires only a day or two of classroom training [2]. This TPI training can be taught to a team, software-related or not, to get them off and running with the process as soon as possible. Please note that since the TPI training does not include all of the original, software-specific principles of Personal Software Process (PSP) [3], software teams are encouraged to take the PSP training sometime in the near future to accelerate their path towards realizing the full benefits of TSP.

All the performance data shown in the figures are from real NAVAIR teams applying TSP or TxP while producing their products or providing their services.

What is TxP?

While TSP is a process containing specific activities that a software team would follow to produce high-quality products in the domain of software, TxP is a set of generic activities (Table 1) that can be tailored to create a process a team would follow to produce high-quality products in the domain of "X." As an example, a system integration test team would use TxP to create the Team Test Process (TTP), and a requirements team would use TxP to create the Team Requirements Process (TRP).

The Path to Applying TxP

While software teams have the option to take two weeks of PSP training and immediately become familiar with how to develop software using PSP and TSP methodologies, other teams outside of software do not have this training available. Some of the metrics applied by TSP teams were identified only after the SEI had analyzed thousands of sets of process data from PSP-practicing individuals. They analyzed the patterns in the process data to discern what differentiated those process data which led to high-quality products from those that resulted in low-quality products. An example of one of these metrics is the appraisal to failure ratio (A/FR), which is calculated by dividing appraisal costs (time spent in design and code reviews) by failure costs (time spent in compile and test). The SEI found that programs with A/FRs greater than 2 have significantly fewer defects discovered in unit testing than those programs with A/FRs less than 2. This is important, since fewer defects found in unit testing usually means fewer defects in the product delivered to the customer. Instead of waiting for SEI to compile enough data to repeat this kind of analysis for the system integration test domain, a system integration test team could use TxP as a checklist for what abilities that team needs to stand up so that it can maximize its chances of doing system integration testing as well as a TSP team does software development.

Even with the TxP checklist in their hands, a team cannot simply 'fill in the blanks' on day one and stand up their Team Test Process (TTP). Some abilities, like planning on how many mistakes will be made by the team in producing a test procedure, can only be performed after the team has determined a) which mistakes count in the domain of system integration testing, b) what units to use to measure the size of a test procedure, and c) how many mistakes

TxP Planning Activities
- Project and Management Objectives
- Team Goals and Roles
- Project Strategy and Support
- Overall Plan
- Planned sizes and rates used to compute times
- Quality Preparation
- Planned Defects Injected/Removed
- Planned quality indicator values are acceptable
- Balanced Plan
- Project Risk Analysis
- Launch Report Preparation
- Management Review
- Launch Postmortem

TxP Working Activities
- Logging time
- Logging defects
- Tracking EV
- Using PROBE in Planning phase
- Entering actual sizes in Postmortem phase
- Defining Defect Types
- Using Review checklists
- Holding periodic team meetings
- Following an agenda during team meetings
- Performing/reporting on assigned roles
- Reviewing action items
- Reviewing assigned goals and risks
- Maintaining project plan and workbook

TxP Analyzing Activities
- Evaluate plan vs. actual schedule hours
- Evaluate plan vs. actual component hours
- Evaluate plan vs. actual component sizes
- Evaluate team performance vs. goals and quality plan
- Evaluate plan vs. actual quality of components
- Update planning data for schedule hours
- Update planning data for lifecycle time-in-phase %s
- Update planning data for productivity rates
- Update planning data for defect densities
- Update planning data for defect rates and yields
- Update planning data for quality indicator thresholds

Table 1. TxP Activities list

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 2014		**00-00-2014 to 00-00-2014**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Evolving Postmortems as Teams Evolve Through TxP	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Air Systems Command (NAVAIR),Code 414300D,STOP 6308, 1900 N. Knox Road,China Lake,CA,93555-6106**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
TxP was created by NAVAIR to help non-software teams define their own customized team processes, similar to the TSP process enjoyed by software teams. It takes time for a team to define a customized team process, and during this time, postmortem analysis needs to evolve as the team?s customized process evolves. These postmortems will give the team the insight required to see where it has improved, and where it needs to focus its future improvement efforts on its way to standing up its customized team process.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**4**	

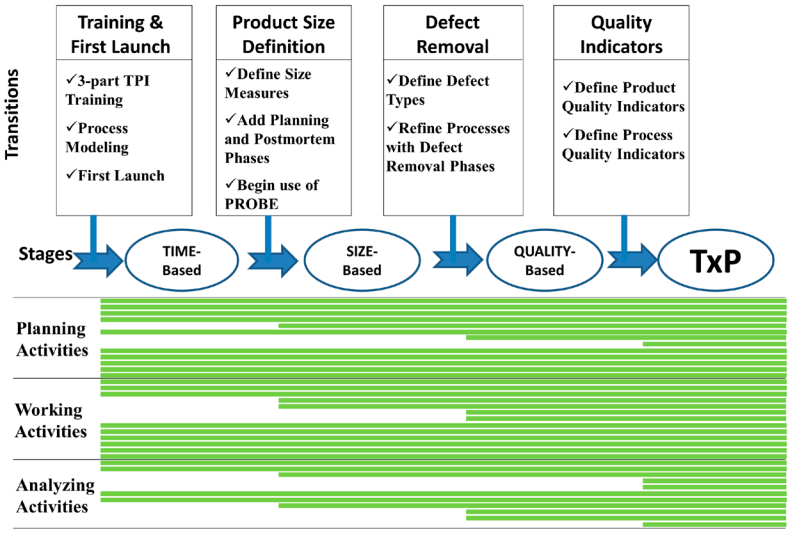


Figure 1. TxP Stages

the team makes when producing test procedures. While software teams come out of the gate with a set of orthogonal defect types identified by Watts Humphrey [4], a system integration test team will have to spend significant time logging mistakes of all kinds before being able to determine which types of mistakes should be logged. Likewise, while software teams may say with certainty that they measure the size of their software products in lines of code, the size measure for a test procedure may not be so obvious. Is it best measured in terms of the number of 'verifies' in the procedure or the number of steps in the procedure? Only after data has been logged, collected, and analyzed by the team, will the team understand which the better size measure is. Finally, only after the team has established which kinds of mistakes should be logged, and agreed on how they will measure the size of a test procedure, will they be able to measure the number of mistakes made. This kind of dependency between the activities causes a team that is starting from scratch (with only TPI training) to have to pass through a number of stages (Figure 1) before being able to completely 'fill in the blanks' and stand up its own customized TxP process.

Evolving Postmortems

Even though a team has not completely defined its customized team processes, it still has a lot of data that can be analyzed to help the team perform its job better in the next cycle. Showing a team how it has improved in some aspect is a very strong motivating factor to help the team understand how easily improvement can be achieved, and to help it embrace process improvement. Postmortems are important in sustaining a team's interest in process improvement. As a team evolves and begins to use more complex plans, more extensive postmortems become possible.

Even PSP/TSP-trained software teams have evolving postmortems. When a team is first introduced to PSP/TSP, there is a huge paradigm shift for the team members to adjust to. Even when the team is sold on TSP as the better way to develop software, there is only so much change that the team can handle and still perform its job of developing software for its customers. Then, as the team gets comfortable and more consistent at performing the primary activities of TSP (e.g., launching, logging time, logging defects, tracking progress, analyzing their data), they begin to look at executing these fundamental activities in more effective ways. The AV-8B Joint System/Software Support Activity declared TSP as their organizational standard for developing software in 2002 and were still evolving its launches and postmortems to become more efficient and effective in 2008 [5].

NAVAIR has developed a standard set of postmortem topics that can be piecewise-introduced as a team gains experience and progresses through the TxP stages on its way to defining its customized team processes. This allows a team to show management that the team is improving its performance in some aspect, regardless of whether the team is just starting with TPI, or well on its way to standing up its customized team processes. These postmortem topics are domain-independent and can be used across an organization as a standard reporting mechanism. This standardization allows organizational managers to become more familiar with the charts presented in the postmortem out-briefs by the organization's various teams, which, in turn, empowers the managers to question the teams more thoroughly on why the charts show what they show.

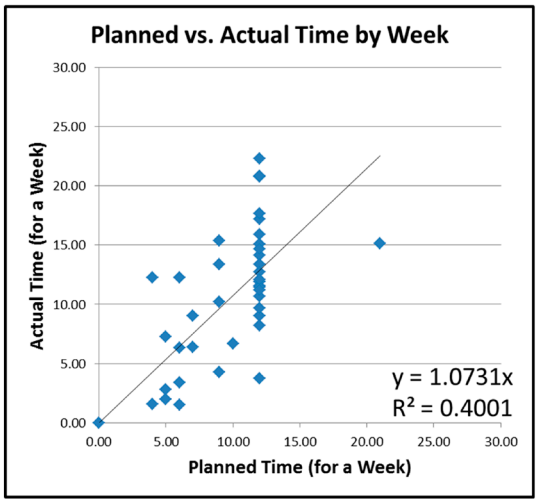
Time-Based Postmortems

The team's most consistent data at first will be time log entries. These entries contain who worked on what task, for how long, and on what date: e.g., Jaime worked on the design of the database user interface for 15 minutes on Thursday. A time log entry is made every time any individual logs any time to any task: e.g., Jaime may have half a dozen entries logged to the database user interface design task that reflect her stolen moments during the week to make progress on that design. With this information, a team can analyze the planned versus actual weekly time spent by individuals on the project (a.k.a. task time). As seen in Figure 2, the analysis can even focus on determining the average actual time logged on only fulltime planned weeks. In this example, 12 planned hours was fulltime so that weeks that contain holidays and leave days do not drive the average actual time per week down artificially. With this insight, individuals will be more attune to their personal weekly task time and will be better prepared to understand how much work they should be able to sign up for during the next cycle's planning session. The scatter chart also provides a quick check to ensure that the team member is maintaining their scheduled hours so that they do not show themselves logging time on a week they planned to be gone (this would be a data point up the left side of the chart), or being gone on a week where they planned to be working: that data point would be along the bottom of the chart.

By looking at the planned versus actual time spent on components (e.g., a user interface, a section of a requirements document, a test procedure), the team can understand its accuracy at estimating the time to create those components. The left chart in Figure 3 provides the team with general and specific information about their estimates. In this case, the value of 0.5018 in the upper left of the chart indicates that the team is only using half of the time they thought they would need, and the data points that land in the upper left and lower right regions of the chart identify specific components whose planned and actual times varied greatly from each other. The team can then discuss these components to understand what made them have actual times so different from their planned times.

As an example of this discuss, concerning the case shown in Figure 3, the team found that two data points (one in the upper left red area and one in the lower right) were Flight Testing-type components. These were two from a group of 291 Flight Testing-type components and were the only two components that were significantly off in their planned times. The team concluded that these two components were simply miss-estimated, and, while the team is still striving to improve its ability to estimate Flight Testing-type components, it can understand when one in a hundred estimates is just plain wrong. The third component (in the lower right red field) belonged to Activity D, which was a very volatile topic with many changing requirements. In this case, this component became much simpler due to a changing requirement but its estimate was not changed to reflect that shift.

By studying the planned versus actual time by type of component, the right chart in Figure 3, the team will get insight into its estimating ability by the various kinds of components it works on. In Figure 3, we see that the team is doing a great job on estimating the time to provide software-in-the-loop (SIL) testing, and landing near the mark when it comes to flight testing and ground testing, but all this is lost in the weeds when combined with the massive 100% or greater errors being realized in estimating activities A, B, C, and D. By showing these categories separately, the team can recognize its accomplishments in the first three categories mentioned and identify the need to improve the way they are estimating activities A, B, C, and D.



Planned Time	Count	Avg Actual Time	% of PT
0	1	0.0	
1	0		
2	0		
3	0		
4	2	6.9	173%
5	3	4.0	80%
6	4	5.9	98%
7	2	7.7	110%
8	0		
9	4	10.8	120%
10	1	6.7	67%
11	0		
12	21	13.5	113%
13	0		
14	0		
15	0		
16	0		
17	0		
18	0		
19	0		
20	0		
21	1	15.2	72%
22	0		

Figure 2. Task Time Charts

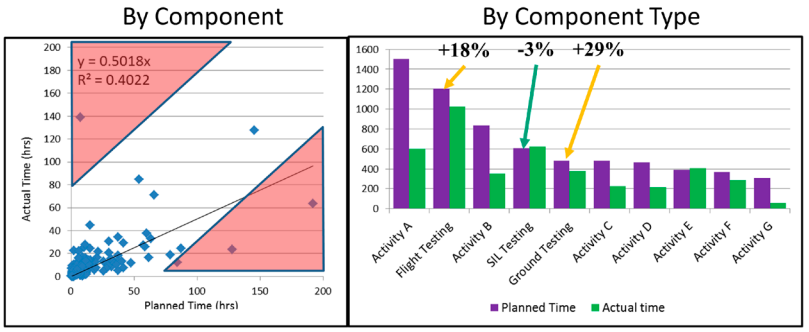


Figure 3. Time By Component Charts

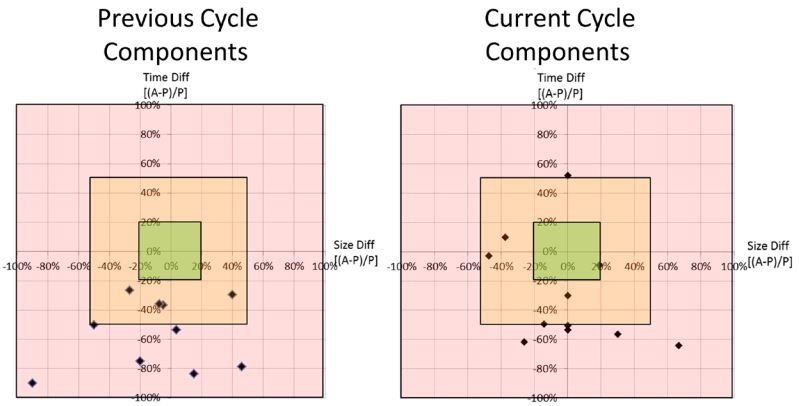


Figure 4. Size and Time Estimation Errors by Component

Size-Based Postmortems

Once the team has determined how they will measure the sizes of their various types of components and has begun to record planned and actual size data on components they have produced, then additional analysis becomes available to them. Just as TSP-trained individuals can compute their personal productivity rates, individuals on a team that logs its time and size data can compute their personal productivity rates in whatever-units-they-are-counting-with per hour. In addition, the team can utilize scatter charts, like those shown in Figure 4, to understand whether their ability to estimate times and sizes is improving from cycle to cycle.

In the example shown, all the data points are along the bottom half of the chart, showing that this team's previous cycle had a trend of consistently overestimating the time to produce a component. Along with that, the team can see that its ability to estimate sizes was mostly in the +/- 50% range, with half of those within +/- 20%. After completing the current cycle, the team can see that the improvements it made in how it estimates time have made a positive difference in that the average time estimate error has moved closer to zero. Their size estimating ability has not improved, still mostly in the +/- 50% range with half of those within +/- 20%. Now that they have made a significant improvement in their time difference errors, they can look to see if they should turn their attention to the size difference errors next, or whether they need to continue to work on improving their time difference errors.

Quality-Based Postmortems

Once consistent mistake-related data is available for analysis, e.g., mistake log entries and mistake type standards, then all kinds of metrics can be quantified concerning the quality of the product as well as the quality of the process. Metrics such as defect injection rates by phase, indicating how frequently they are making mistakes while they are logging time to a certain phase of the process, help the team to understand where the mistakes are being made in the process. With that information, the team can then discuss if there is anything they can change on how they perform that phase of the process to reduce the rate at which mistakes are being made.

Watts Humphrey injected design and code review phases into the software development process in support of the PSP principle that if the mistake is fixed as soon as possible after it is made,

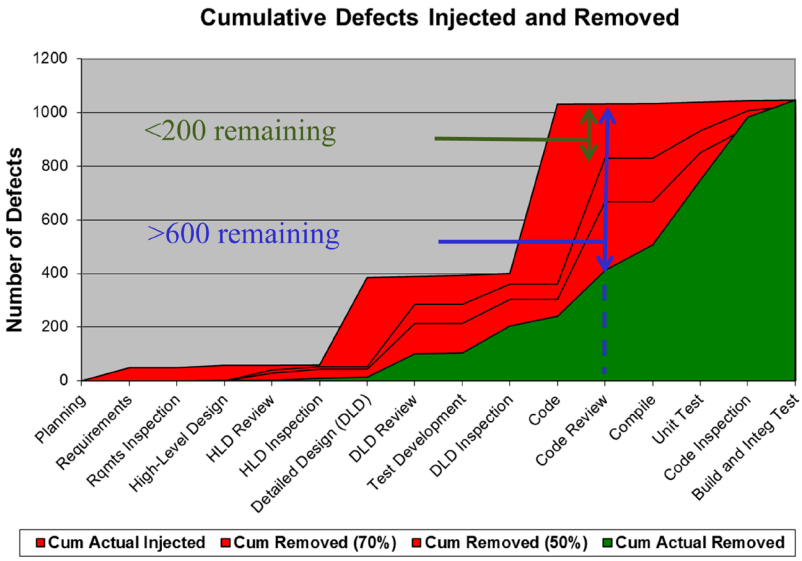


Figure 5. Defects Injected and Removed

and fixed by the person who made it, then the cost of fixing that mistake would be minimized [4]. Building a chart similar to that shown in Figure 5, the team can see where, in their process, they are making their mistakes, and where they are fixing their mistakes. In studying this type of chart, teams may recognize the need to add additional phases to their current process, or remove phases that are not providing added value, or see where they are performing better or worse than the planned performance. Comparison lines can be added to show the team where they planned on finding their mistakes to allow the team to recognize where in the process they are struggling most to meet the plan. In this example, the team needs to focus on improving its performance during code review.

TxP-Based Postmortems

Once the team has been operating at the Quality stage for some time, then analysis can start on the accumulated data to try and identify leading indicators of the quality of the product, i.e., the equivalent of an A/FR for a test procedure. This analysis can likewise be applied to find leading indicators for the quality of the process. In addition, this analysis can lead to identifying acceptable thresholds for rates and ratios which can be used to assess the quality of a plan (Figure 6 shows some of the quality rates and ratios used in TSP). Identifying these leading indicators and acceptable thresholds allows the team even more insight into ways of improving the quality of their plans, processes, and (most importantly) products. Only after the team can identify what level of process performance leads to a quality product can they then, with certainty, compare planned and actual values of these leading indicators, rates, and ratios and know which values are more desirable.

Summary

As a team's process evolves from TPI to a customized team process, the postmortem analysis of their data needs to evolve too. The focus of the analysis should be on what is value-added to the team and that analysis should help them to identify what progress they have made so far and where they need to continue to focus their attention. These analysis efforts should peak the team's interest in process improvement, but will definitely lead to improvements in planning, product quality, and communication with management.

REVIEW RATES (LOC/hr)		
Phase	Plan	Actual
DLD Review	336	829
DLD Inspection	71	136
CODE Review	147	266
CODE Inspection	60	62

RATIOS		
Phase	Plan	Actual
DLD Review / DLD Ratio	0.36	0.27
DLD / Code Ratio	0.82	0.59
Code Review / Code Ratio	0.33	0.20
Compile Defect Density (defects/KLOC)	0.00	3.15
Unit Test Defect Density (defects/KLOC)	8.86	7.81

Cost of Quality (COQ)		
Topic	Plan	Actual
% Appraisal COQ	36.8%	30.8%
% Failure COQ	19.9%	26.1%
Appraisal / Failure Ratio (AFR)	1.85	1.18

Figure 6. TSP Quality Indicators

ABOUT THE AUTHOR

Brad Hodgins is a TSP/PSP coach and instructor for NAVAIR at China Lake, California, where he coaches engineering teams in the development of high quality aviation products for the U.S. Navy and Marine Corps. He is a NAVAIR Associate Fellow and has been awarded a U.S. Navy patent for the Learning Applying Mastering Perfecting (LAMP) model for team process implementation, evaluation, and improvement. His MS in Computer Science is from Colorado Technical University.

NAVAIR
Code 414300D
STOP 6308, 1900 N. Knox Road
China Lake, CA 93555-6106
Phone: 760-939-0666
E-mail: bradley.hodgins@navy.mil

REFERENCES

1. NAVAIR is the Naval Air Systems Command. NAVAIR procures, develops, tests, and supports Naval aircraft, weapons, and related systems. For more information about NAVAIR, go to <www.navair.navy.mil>
2. Schwalb, Jeff and Brad Hodgins. Broadening the Ability to Train and Launch Effective Engineering and Service Teams. 1 September 2011. <http://www.sei.cmu.edu/tspsymposium/past-proceedings/2011/Broadening-the-Ability-to-Train.pdf>.
3. Personal Software Process (PSP) is a data-driven method of developing software that gives the software engineer insight as to where their process needs to be improved for them to produce a higher quality software product. PSP also helps to improve their ability to estimate the labor and calendar time required to produce that software product. A more thorough description of PSP can be found at <http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=5283>
4. Humphrey, Watts S. A Discipline for Software Engineering. Reading, MA: Addison-Wesley, 1995.
5. Rickets, Chris and Brad Hodgins. How TSP Implementation Has Evolved at AV-8B. 1 May 2008. <http://www.ieee-stc.org/proceedings/2008/pdfs/BH1997.pdf>.